RIO-47100

COMMAND REFERENCE

Manual Rev. beta1

By Galil Motion Control, Inc.

Galil Motion Control, Inc.
3750 Atherton Road
Rocklin, California 95765
Phone: (916) 626-0101
Fax: (916) 626-0102
Internet Address: support@galilmc.com
URL: www.galilmc.com

Rev 8/20/07

#... 5
$... 6
& | ... 7
( ) ... 8
; ... 9
[ ] ... 10
+ - * / % ... 11
, =, <=, >=, <> ... 12
= ... 13
` (Ascii 96) ... 14
AA ... 15
AB ... 16
@ABS[n] ... 17
@ACOS[n] ... 18
AF ... 19
AI ... 20
@AN[n] ... 21
AO ... 22
@AO[n] ... 23
AQ ... 24
@ASIN[n] ... 25
AT ... 26
@ATAN[n] ... 27
#AUTO ... 28
#AUTOERR ... 29
AZ ... 30
BK ... 31
BN ... 32
BP ... 33
BV ... 34
CB ... 35
CF ... 36
CI ... 37
CL ... 38
#CMDERR ... 39
@COM[n] ... 40
@COS[n] ... 41
CW ... 42
DA ... 43
DB ... 44
DH ... 45
DL ... 46
DM ... 47
DR ... 48
ED ... 50
ELSE ... 51
EN ... 52
ENDIF ... 53
EO ... 54
@FRAC[n] ... 55
HS ... 56
HX ... 57
IA ... 58
ID ... 59
IF ... 60
IH ... 61
II ... 63
IL ... 64
IN ... 65
@IN[n] ... 66
#ININTn ... 67

@INT[n] 68
IQ 69
JP 70
JS 71
KD 72
KI 73
KP 74
LA 75
{control}L{control}K 76
LL 77
LS 78
LV 79
LZ 80
MA 81
MB 82
MD 84
MG 85
MS 86
MW 87
NO (' apostrophe also accepted) 88
OB 89
OF 90
OP 91
@OUT[n] 92
PC 93
PS 94
PW 95
QD 96
QR 97
QU 98
QZ 99
RA 100
RC 101
RD 102
RE 103
REM 104
RI 105
@RND[n] 106
RS 107
{control}R{control}S 108
{control}R{control}V 109
SA 110
SB 111
@SIN[n] 112
SL 113
SM 114
@SQR[n] 115
@TAN[n] 116
TB 117
TC 118
#TCPERR 120
TE 121
TH 122
TI 123
TIME* 124
TR 125
TZ 126
UL 127
VF 128
WH 129
WT 130
XQ 131

ZC 132
ZD 133
ZS 134

# #

FUNCTION: Label (subroutine)

DESCRIPTION:
The # operator denotes the name of a program label (for example #Move). Labels can be up to seven characters long and are often used to implement subroutines or loops. Labels are divided into (a) user defined and (b) automatic subroutines. User defined labels can be printed with LL and the number of labels left available can be queried with MG _DL. The maximum number of labels available is 62. The automatic subroutines include #CMDERR, #TCPERR, #ININT, #AUTO, and #AUTOERR.

ARGUMENTS: #nnnnnnn where
nnnnnnn is a label name up to seven characters

USAGE:

Default Value	-		
In a Program	Yes	Default Format	-
Command Line	No		

OPERAND USAGE:
RELATED COMMANDS:

LL	List labels
DL	Labels left
JP	Jump statement
JS	Jump subroutine

EXAMPLES:

```
i=0
#Loop
JP#Loop, i<10          ;'jump to #Loop until i becomes 10
EN
```

$

FUNCTION: Hexadecimal

DESCRIPTION:
The $ operator denotes that the following string is in hexadecimal notation

ARGUMENTS: $nnnnnnnn.mmmm
n is up to eight hexadecimal digits (denoting 32 bits of integer)
m is up to four hexadecimal digits (denoting 16 bits of fraction)

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
+ - * / % Multiply (shift left)
+ - * / % Divide (shift right)
MG {$8.4} Print in hexadecimal

EXAMPLES:
```
x = $7fffffff.0000                  ;'store 2147483647 in x
y = x & $0000ffff.0000           ;'store lower 16 bits of x in y
z = x & $ffff0000.0000 / $10000         ;'store upper 16 bits of x in z
```

& |

FUNCTION: Bitwise Logical Operators AND and OR

DESCRIPTION:
The operators & and | are typically used with IF, JP, and JS to perform conditional jumps; however, they can also be used to perform bitwise logical operations.

ARGUMENTS: n & m or n | m where
n and m are signed numbers in the range -2147483648 to 2147483647.
For IF, JP, and JS, n and m are typically the results of logical expressions such as (x > 2)

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@COM[n] Bitwise complement
IF If statement
JP Jump statement
JS Jump subroutine

EXAMPLES:
```
IF (x > 2) & (y = 4)
  MG "true"
ENDIF                    ;'x must be greater than 2 and y equal to 4
                       ;'for the message to print
:MG 1 | 2
3.0000
:          ;'Bitwise operation:  01 OR 10 is 11 = 3
```

( )

FUNCTION: Parentheses (order of operations)

DESCRIPTION:
The parentheses denote the order of math and logical operations. Note that the controller DOES NOT OBEY STANDARD OPERATOR PRECEDENCE. For example, multiplication is NOT evaluated before addition. Instead, the controller follows left-to-right precedence. Therefore, it is recommended to use parenthesis as much as possible.

ARGUMENTS: (n) where
n is a math (+ - * /) or logical (& |) expression

USAGE:

Default Value	-		
In a Program	Yes	Default Format	-
Command Line	Yes		

OPERAND USAGE:
RELATED COMMANDS:
+ - * / %
Math Operators
& |
Logical Operators

EXAMPLES:

```
:MG 1 + 2 * 3
9.0000
:MG 1 + (2 * 3)
7.0000
```

;

FUNCTION: Semicolon (Command Delimiter)

DESCRIPTION:
The semicolon operator allows multiple Galil commands to exist on a single line. It is used for the following three reasons:
(1) To put comments on the same line as the command (BGX ;'begin motion)
(2) To compress DMC programs to fit within the program line limit (40 characters x 200 lines)
(3) To give higher priority to a thread. All commands on a line are executed before the thread scheduler switches to the next thread. (4 threads total)

ARGUMENTS: n; n; n; where
n is a Galil command

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
NO (' apostrophe also accepted)
comment

EXAMPLES:
```
SB1;WT500;CB1;'multiple commands separated by semicolons
```

[ ]

FUNCTION: Square Brackets (Array Index Operator)

DESCRIPTION:
The square brackets are used to denote the array index for an array, or to denote an array name. (They are also used to designate the argument to a function, such as @ABS[n].)

ARGUMENTS: mmmmmmmm[n] where
mmmmmmmm is the array name
n is the array index and is an integer between 0 and 399

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
DM
Dimension Array
QU
Print/Upload Array

EXAMPLES:
```
DM A[50]          ;'define a 50 element array
A[0] = 3          ;'set first element to 3
MG A[0]                ;'print element 0
QU A[]                ;'print entire array
```

+ - * / %

FUNCTION: Math Operators

DESCRIPTION:
The addition, subtraction, multiplication, division, and modulus operators are binary operators (they take two arguments and return one value) used to perform mathematical operations on variables, constants, and operands.

ARGUMENTS: (n + m) or (n - m) or (n * m) or (n / m) or (n % m) where
n and m are signed numbers in the range -2147483648 to 2147483647

USAGE:

Default Value	-		
In a Program	Yes	Default Format	-
Command Line	Yes		

OPERAND USAGE:
RELATED COMMANDS:
( )
Parenthesis

EXAMPLES:

```
:x =((1+(2*3))/7)-2          ;'assign -1 to x
:MG 40 % 6                   ;'integer remainder of 40 divided by 6
  4.0000
```

, =, <=, >=, <>

FUNCTION: Comparison Operators

DESCRIPTION:
The comparison operators are as follows:
< less than
> greater than
= equals
<= less than or equal
>= greater than or equal
<> not equals
These are used in conjunction with IF, JP, JS, ( ), &, and | to perform conditional jumps. The result of a comparison expression can also be printed with MG or assigned to a variable.

ARGUMENTS: (n < m) or (n > m) or (n = m) or (n <= m) or (n >= m) or (n <> m) where
n and m are signed numbers in the range -2147483648 to 2147483647

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
( ) Parentheses
IF If statement
JP Jump
JS Jump subroutine

EXAMPLES:
```
IF(x > 2) & (y = 4)
MG "true"
ENDIF          ;'x must be greater than 2 and y equal to 4 for
          ;'the message to print
```

=

FUNCTION: Equals (Assignment Operator)

DESCRIPTION:
The assignment operator is used for three reasons:
(1) to define and initialize a variable (x = 0) before it is used
(2) to assign a new value to a variable (x = 5)
(3) to print a variable or array element (x= which is equivalent to MG x). MG is the preferred method of printing.

ARGUMENTS: mmmmmmmm = n where
mmmmmmmm is a variable name and n is a signed number in the range -2147483648 to 2147483647

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
MG
Print Message

EXAMPLES:
```
:x=5
:x=          ;'define and initialize x to 5
5.0000
:MG x          ;'print x two different ways
5.0000
```

` (Ascii 96)

FUNCTION: Line Continuation Character

DESCRIPTION:
Allows a command in an application program to extend beyond the confines of the maximum line length of 40 characters. This is especially useful for code compression, long MG statements, or multiple conditions in an IF,JP or JS statement.

ARGUMENTS: none

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
MG Print Message

EXAMPLES:
```
IF((var100=1000)&(var101=50));MG"GO";EL`
SE;MG"STOP";ENDIF;
```

AA

FUNCTION: After Analog Trippoint

DESCRIPTION:
AA is a trippoint that halts program execution until a voltage on a particular analog input has been reached

ARGUMENTS: AA m,n where
m is the analog input (0-7)
n = the voltage which ranges from 0-5V

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
@AN[x] Function to query the voltage on an analog input

EXAMPLES:

AB

FUNCTION: Abort

DESCRIPTION:
AB (Abort) aborts the application programs including any threads

ARGUMENTS: AB

USAGE:
In a Program Yes
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
"HX" Halt Execution

EXAMPLES:

@ABS[n]

FUNCTION: Absolute value

DESCRIPTION:
Takes the absolute value of the given number. Returns the value if positive, and returns -1 times the value if negative.

ARGUMENTS: @ABS[n] where
n is a signed number in the range -2147483647 to 2147483647

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@SQR[n] Square Root

EXAMPLES:
```
:MG @ABS[-2147483647]
2147483647.0000
```

@ACOS[n]

FUNCTION: Inverse cosine

DESCRIPTION:
Returns in degrees the arc cosine of the given number.

ARGUMENTS: @ACOS[n] where
n is a signed number in the range -1 to 1.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ASIN[n] Arc sine
@SIN[n] sine
@ATAN[n] Arc tangent
@COS[n] Cosine
@TAN[n] Tangent

EXAMPLES:
```
:MG @ACOS[-1]
180.0000
:MG @ACOS[0]
90.0000
:MG @ACOS[1]
0.0001
```

AF

FUNCTION: Analog Feedback Select

DESCRIPTION:
The Analog Feedback (AF) command is used to select which analog input channel will be used for feedback if a control loop is desired.

ARGUMENTS: AF n,n where
n = -1 (default) non selected
n = 0-7 selects the analog input channel for feedback
n = ? Returns the input channel

USAGE:
Default Value -1,-1
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
_AFx contains the analog input channel selected

RELATED COMMANDS:
AQ Analog Configuration
AZ Analog Output Select
PS Control Loop Setpoint

EXAMPLES:
```
AF0
AZ0
KP1
KD10
KI0.5
PS2.5
```

AI

FUNCTION: After Input

DESCRIPTION:
The AI command is used in programs to wait for a specified input pattern. If the parameter entered is positive, it waits for that input to go high. If negative, it waits for that input to go low.

ARGUMENTS: AI m1& m2& m3& where
mn is an integer in the range 0 to 15 decimal. When mn is positive, the RIO will wait for the input to go high. When mn is negative, the ler will wait for the input to go low.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
@IN[n] Function to read inputs
II Input interrupt

EXAMPLES:
```
#A                  Begin Program
AI 7&15&-1&-12         Wait until inputs 7 & 15 are high, and inputs 1 & 12 are
low
MG "DONE"          Send message 'DONE' when conditions are satisfied
EN                   End Program
HINT: The AI command actually halts execution until specified input(s) is at
desired logic level.  Use the conditional Jump command (JP) or input interrupt
(II) if you do not want the program sequence to halt.
```

@AN[n]

FUNCTION: Analog Input Query

DESCRIPTION:
The @AN[n] command is used to query the value of an Analog Input.

ARGUMENTS: @AN[n] where
n is the I/O number assigned to a particular analog input pin (0-7)

USAGE:
In a Program Yes
Command Line Yes
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"AO" Analog Output

EXAMPLES:
```
MG@AN[0]      Displays status of Analog input 0
Temp=@AN[0]    Sets variable Temp to the value of Analog input 0
```

AO

FUNCTION: Analog Output

DESCRIPTION:
The AO command sets the analog output voltage on an analog output pin. It is also used to set the analog output voltage on ModBus devices over Ethernet

ARGUMENTS: AO m, n where
m is the I/O number assigned to a particular analog output pin (0-7)

m can also be the I/O number of a ModBus device that is calculated using the following equations:
m = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)
HandleNum is the handle specifier from A to C (1 to 3).
Module is the position of the module in the rack from 1 to 16.
BitNum is the I/O point in the module from 1 to 4.

n = the voltage which ranges from 0-5V.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"SB" Set Bit
"CB" Clear Bit

EXAMPLES:

@AO[n]

FUNCTION: Analog Output Query

DESCRIPTION:
The @AO[n] command is used to query the value of an Analog Output.

ARGUMENTS: @AO[n] where
n is the I/O number assigned to a particular analog output pin (0-7)

USAGE:
In a Program Yes
Command Line Yes
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"AO" Analog Output

EXAMPLES:
```
MG@AO[0]      Displays status of Analog output 0
Temp=@AO[0]    Sets variable Temp to the value of Analog output 0
```

AQ

FUNCTION: Analog Configuration

DESCRIPTION:
The Analog Configuration (AQ) command is used to set the type of analog input.

ARGUMENTS: AQ n,m where
n is an even integer from 0-6 that represents the analog input channel
m = 0 (default) single ended inputs
m = 1 differential inputs (next odd channel is consumed)

USAGE:
Default Value n,0
In a Program Yes Default Format 1.0000
Command Line Yes

OPERAND USAGE:
_AQn

RELATED COMMANDS:
@AN[n] Read Analog Input

EXAMPLES:
```
AQ 2,1;          Sets Analog input 2 and 3 to be differential inputs
```

@ASIN[n]

FUNCTION: Inverse sine

DESCRIPTION:
Returns in degrees the arc sine of the given number.

ARGUMENTS: @ASIN[n] where
n is a signed number in the range -1 to 1.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ACOS[n] Arc cosine
@SIN[n] sine
@ATAN[n] Arc tangent
@COS[n] Cosine
@TAN[n] Tangent

EXAMPLES:
```
:MG @ASIN[-1]
-90.0000
:MG @ASIN[0]
0.0000
:MG @ASIN[1]
90.0000
```

AT

FUNCTION: At Time

DESCRIPTION:
The AT command is a trippoint which is used to hold up execution of the next command until after the specified time has elapsed. The time is measured with respect to a defined reference time. AT 0 establishes the initial reference. AT n specifies n msec from the reference. AT -n specifies n msec from the reference and establishes a new reference after the elapsed time period.

ARGUMENTS: AT n where
n is a signed integer in the range 0 to 2 Billion
n = 0 defines a reference time at current time
positive n waits n msec from reference
negative n waits n msec from reference and sets new reference after elapsed time period
(AT -n is equivalent to AT n; AT 0)

USAGE:
In a Program Yes
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
The following commands are sent sequentially
AT 0         Establishes reference time 0 as current time
AT 50         Waits 50 msec from reference 0
AT 100         Waits 100 msec from reference 0
AT -150         Waits 150 msec from reference 0 and sets new reference at 150
AT 80         Waits 80 msec from new reference (total elapsed time is 230 msec
from previous command)
```

@ATAN[n]

FUNCTION: Inverse tangent

DESCRIPTION:
Returns in degrees the arc tangent of the given number.

ARGUMENTS: @ATAN[n]
n is a signed number in the range -2147483647 to 2147483647

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ASIN[n] Arc sine
@SIN[n] sine
@ACOS[n] Arc cosine
@COS[n] Cosine
@TAN[n] Tangent

EXAMPLES:
```
:MG @ATAN[-10]
-84.2894
:MG @ATAN[0]
0.0000
:MG @ATAN[10]
84.2894
```

#AUTO

FUNCTION: Subroutine to run automatically upon power up

DESCRIPTION:
#AUTO denotes code to run automatically when power is applied to the controller, or after the controller is reset. When no host software is used with the controller, #AUTO and the BP command are required to run an application program on the controller.

ARGUMENTS:
USAGE:
In a Program Yes
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
BP Burn program
EN End program

EXAMPLES:
```
#AUTO                  ;'Start on powerup
SB1                   ;'Set bit 1
WT500                  ;'Wait 500msec
CB1                   ;'Clear bit 1
JP#AUTO                    ;'Jump back to #AUTO
EN                   ;'EN to end all programs
```

#AUTOERR

FUNCTION: Automatic subroutine for notification of EEPROM checksum errors

DESCRIPTION:
#AUTOERR will run code upon power up if data in the EEPROM has been corrupted. The EEPROM is considered corrupt if the checksum calculated on the bytes in the EEPROM do not match the checksum written to the EEPROM. The type of checksum error can be queried with _RS

ARGUMENTS:
USAGE:
In a Program Yes
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
_RS
Checksum error code
EN
End program

EXAMPLES:
```
#AUTO
  WT 2000
  MG "AUTO"
  JP#AUTO
EN
#AUTOERR
  WT500
  MG "AUTOERR ", _RS
EN
NOTE: Use EN to end the routine
```

AZ

FUNCTION: Analog Output Select

DESCRIPTION:
Selects the Analog Outputs used for control loop

ARGUMENTS: AZ n,n where
n = -1 (default) none selected
n = 0 through 7 selects the analog output channel for control

USAGE:
Default Value -1,-1
In a Program Yes Default Format 1.0000
Command Line Yes

OPERAND USAGE:
_AZ

RELATED COMMANDS:
@AN[n] Read Analog Input
AF Analog Feedback
PS Control Loop Setpoint

EXAMPLES:
```
AZ0,1;  'set analog outputs 0 and 1 as control loop outputs
```

BK

FUNCTION: Breakpoint

DESCRIPTION:
For debugging. Causes the controller to pause execution of the given thread at the given program line number (which is not executed). All other threads continue running. Only one breakpoint may be armed at any time. After a breakpoint is encountered, a new breakpoint can be armed (to continue execution to the new breakpoint) or BK will resume program execution. The SL command can be used to single step from the breakpoint. The breakpoint can be armed before or during thread execution.

ARGUMENTS: BK n,m where
n is an integer in the range 0 to 199 which is the line number to stop at. n must be a valid line number in the chosen thread.
m is an integer in the range 0 to 3. The thread.

USAGE:
Default Value of m 0
In a Program No
Command Line Yes

OPERAND USAGE:
_BK will tell whether a breakpoint has been armed, whether it has been encountered, and the program line number of the breakpoint:
= -LineNumber: breakpoint armed
= LineNumber: breakpoint encountered
= -2147483648: breakpoint not armed

RELATED COMMANDS:
SL Single Step
TR Trace

EXAMPLES:
```
BK 3          Pause at line 3 (the 4th line) in thread 0
BK 5          Continue to line 5
SL          Execute the next line
SL 3          Execute the next 3 lines
BK          Resume normal execution
```

BN

FUNCTION: Burn

DESCRIPTION:
The BN command saves certain board parameters in non-volatile EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The RIO board returns a: when the Burn is complete.

ARGUMENTS:
USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes
PARAMETERS SAVED DURING BURN:

CB	AQ	MA
CW	SB	MS
EO	SM	MD
IA	VF	IQ
LZ	PW	

OPERAND USAGE:
_BN contains the serial number of the unit

RELATED COMMANDS:
"BP" Burn Program
"BV" Burn Variables

EXAMPLES:

BP

FUNCTION: Burn Program

DESCRIPTION:
The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 1 seconds to execute and must not be interrupted. The unit returns a : when the Burn is complete.

ARGUMENTS:
USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"BN" Burn Parameters
"BV" Burn Variables

EXAMPLES:
```
Note: This command may cause the Galil software to issue the following warning
"A time-out occurred while waiting for a response from the controller". This
warning is normal and is designed to warn the user when the controller does not
respond to a command within the timeout period. This occurs because this
command takes more time than the default timeout period. The timeout can be
changed in the Galil software. This warning does not affect the operation of the
board or software.
```

BV

FUNCTION: Burn Variables and Array

DESCRIPTION:
The BV command saves the defined variables and arrays in non-volatile EEPROM memory. This command typically takes up to 2 seconds to execute and must not be interrupted. The unit returns a : when the Burn variables is complete. The RIO can store 126 variables and 4 arrays (total of 400 elements) to non-volatile EEPROM.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"BN" Burn Parameters
"BP" Burn Program

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the controller does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout period. The timeout can be changed in the Galil software. This warning does not affect the operation of the board or software.

CB

EXAMPLES:

CB

FUNCTION: Clear Bit

DESCRIPTION:
The CB command clears a particular output bit, setting the output to logic 0. The CB and SB (Set Bit) instructions can be used to control the state of output lines. The CB command can also be used with modbus devices to toggle off-board outputs.

ARGUMENTS: CB n where
n is an integer corresponding to a specific output to be cleared (0-15 for on-board outputs).
When using Modbus devices, the I/O points of the Modbus devices are calculated using the following formula:

n = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

HandleNum is the handle specifier, labeled with letters from A to C (1 to 3).
Module is the position of the module in the rack from 1 to 16.
BitNum is the I/O point in the module from 1 to 4.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"SB" Set Bit

EXAMPLES:

```
CB 0          Clear output bit 0
CB 1          Clear output bit 1
CB 2          Clear output bit 2
```

CF

FUNCTION: Configure

DESCRIPTION:
Sets the default port for unsolicited messages. The CF command allows the user to configure which port to use for unsolicited messages (messages that come from a program instead of a response to a command). Options include Ethernet Handles A, B, C, or S for the Serial port.

ARGUMENTS: CF n where
n is A (default),B or C for Ethernet handles, and S for the serial port.

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
_CF will return the current port selected for unsolicited responses from RIO. The _CF operand will return a decimal value.

RELATED COMMANDS:
EXAMPLES:

```
CFA          Select Ethernet handle A to return unsolicited responses.
MG_CF          Interrogate configuration
:65.000          Response from _CF showing handle A as default port in terms of
the ASCII character 65 (A)
```

CI

FUNCTION: Configure Communication Interrupt

DESCRIPTION:
The CI command configures a program interrupt based on characters received on the serial communications port. An interrupt causes program flow to jump to the #COMINT subroutine. If multiple program threads are used, the #COMINT subroutine runs in thread 0 and the remaining thread continues to run without interruption. The characters received can be accessed via the internal variables P1CH, P1ST, P1NM, P1CD.

ARGUMENTS: CI n,m

PARAMETER	EXPLANATION
n = 0	Do not interrupt (default)
n = 1	Interrupt on carriage return
n = 2	Interrupt on any character
n = -1	Clear interrupt data buffer

m = 0 Default
m = 1 Enable serial port for CI execution

USAGE:

Default Value	n = 0, m = 0		
In a Program	Yes	Default Format	-
Command Line	Yes		

OPERAND USAGE:
RELATED COMMANDS:

CC	Configure communications
IN	Communication input
MG	Message output

EXAMPLES:

```
CI 1          Interrupt when the key is received
CI 2          Interrupt on a single character received
```

CL

FUNCTION: Control Loop

DESCRIPTION:
The CL command allows the user to select the update rate in milliseconds for the optional control loops. The default update rate is 1msec.

ARGUMENTS: CL n where
n is an integer in the range of 1 to 65535 that represents the update rate in milliseconds.

USAGE:

Default Value	1		
In a Program	Yes	Default Format	-
Command Line	Yes		

OPERAND USAGE:
RELATED COMMANDS:

EXAMPLES:

#CMDERR

FUNCTION: Command error automatic subroutine

DESCRIPTION:
Without #CMDERR defined, if an error occurs (see TC command) in an application program running on the Galil controller, the program (all threads) will stop. #CMDERR allows the programmer to handle the error by running code instead of stopping the program.

ARGUMENTS:
USAGE:
In a Program Yes
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
TC Tell Error Code
_ED Last program line with an error
EN End program

EXAMPLES:
```
#CMDERR                  ;'Command error utility
  MG"Error on line ",_ED
  MG"Error Code ",_TC          ;'Print Error code
  ZS0                    ;'Zero stack
EN1                      ;'End program
NOTE: An application program must be executing for #CMDERR to execute, which
runs in thread 0.
NOTE: Use EN to end the routine
```

@COM[n]

FUNCTION: Bitwise complement

DESCRIPTION:
Performs the bitwise complement (NOT) operation to the given number

ARGUMENTS: @COM[n] where
n is a signed integer in the range -2147483647 to 2147483647.
The integer is interpreted as a 32-bit field.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
& | Logical operators AND and OR

EXAMPLES:
```
:MG {$8.0} @COM[0]
$FFFFFFFF
:MG {$8.0} @COM[$FFFFFFFF]
$00000000
```

@COS[n]

FUNCTION: Cosine

DESCRIPTION:
Returns the cosine of the given angle in degrees

ARGUMENTS: @COS[n] where
n is a signed number in degrees in the range of -32768 to 32767, with a fractional resolution of 16-bit..

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ASIN[n] Arc sine
@SIN[n] Sine
@ATAN[n] Arc tangent
@ACOS[n] Arc cosine
@TAN[n] Tangent

EXAMPLES:
```
:MG@COS[0]
1.0000
:MG@COS[90]
0.0000
:MG@COS[180]
-1.0000
:MG@COS[270]
0.0000
```

CW

FUNCTION: Copyright information / Data Adjustment bit on/off

DESCRIPTION:
The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement for use by the Galil terminal software programs. When turned on, the communication enhancement from the command sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are characters that are returned from a program running on the controller. This command does not affect solicited characters - which are characters that are returned as a response to a command sent from a host PC.

ARGUMENTS: CW n,m where
n is a number, either 0,1 or 2:
0 or ? Causes the controller to return the copyright information
1 Causes the controller to set the MSB of unsolicited returned characters to 1
2 Causes the controller to not set the MSB of unsolicited characters.
m is 0 or 1 (optional)
0 Causes the controller to pause program execution when hardware handshaking disables character transmissions.
1 Causes the controller to continue program execution when hardware handshake disables character transmissions - output characters will be lost.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_CW contains the value of the data adjustment bit. 1 =on, 2 = off

RELATED COMMANDS:
EXAMPLES:
*Note: The CW command can cause garbled characters to be returned by the controller. The default state of the board is to disable the CW command. However, the terminal software may enable the CW command for internal usage. If the board is reset while the Galil software is running, the CW command could be reset to the default value, which creates difficulty for the software. It may be necessary to re-enable the CW command. The CW command status can be stored in EEPROM.

DA

FUNCTION: Deallocate the Variables & Arrays

DESCRIPTION:
The DA command frees the array and/or variable memory space. In this command, more than one array or variable can be specified for memory de-allocation. Different arrays and variables are separated by comma when specified in one command. The * argument deallocates all the variables, and *[0] deallocates all the arrays.

ARGUMENTS: DA c[0],d,etc. where
c[0] - Defined array name
d - Defined variable name
* - Deallocates all the variables
*[0] - Deallocates all the arrays
DA? Returns the number of arrays available on the RIO.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
_DA contains the total number of arrays available. For example, before any arrays have been defined, the operand _DA is 4. If one array is defined, the operand _DA will return 3.

RELATED COMMANDS:
"DM"
Dimension Array

EXAMPLES:
```
  'Cars' and 'Salesmen' are arrays, and 'Total' is a variable.
DM Cars[40],Salesmen[50]         Dimension 2 arrays
Total=70                          Assign 70 to the variable Total
DA Cars[0],Salesmen[0],Total        Deallocate the 2 arrays & variable
DA*[0]                                   Deallocate all arrays
DA *,*[0]                              Deallocate all variables and all arrays
NOTE:  Since this command deallocates the spaces and compacts the array spaces
in the memory, it is possible that execution of this command may take longer
time than a standard command.  Variables and arrays that are deallocated are not
set to zero.  A routine that writes zeros to the array and/or variables should
be created if this is desired.
```

DB

FUNCTION: Deadband

DESCRIPTION:
The DB command selects the deadband where the error must exceed the deadband to execute the control loop.

ARGUMENTS: DB n where
n is a decimal value in the range of 0 to 1.0000V

USAGE:
Default Value 0
In a Program Yes Default Format -----
Command Line Yes

OPERAND USAGE:
_DB

RELATED COMMANDS:
AF Analog Feedback Select
AZ Analot Output Select
CL Control Loop
PS Setpoint

EXAMPLES:

DH

FUNCTION: DHCP Server Enable

DESCRIPTION:
The DH command configures the DHCP or BOOT-P functionality on the controller for Server IP addressing.

ARGUMENTS: DH n where
n = 0 disables DHCP and enables BOOT-P
n = 1 disables BOOT-P and enables DHCP(default)

USAGE:
Default Value 1
In a Program Yes Default Format -----
Command Line Yes

OPERAND USAGE:
_DH returns the state of the DHCP server enable

RELATED COMMANDS:
IA IP Address

EXAMPLES:

DL

FUNCTION: Download

DESCRIPTION:
The DL command transfers a data file from the host computer to the controller. If this command is typed on the command input line of the Galil Terminal, another window will come up asking the user to specify a file to download. If the DL command is used in a terminal utility such as Hyper Terminal, a file will be accepted as a datastream without line numbers. In this case, the file is terminated using {control} Z, {control} Q, {control} D, or \. DO NOT insert spaces before each command.
If no parameter is specified, downloading a data file will clear any programs in the controller board's RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the controller will return a "?". To download a program after a label, specify the label name following DL. The # argument may be used with DL to append a file at the end of the controller program in RAM.

ARGUMENTS: DL n
n = no argument Downloads program beginning at line 0. Erases programs in RAM.
n = #Label Begins download at line following #Label where label may be any valid program label.
n = # Begins download at end of program in RAM.

USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
When used as an operand, _DL gives the number of available labels. The total number of labels is 62.

RELATED COMMANDS:
"UL" Upload

EXAMPLES:
```
DL          Begin Download
SB0          Data
CB2          Data
EN          Data
{control}Z          End download
```

DM

FUNCTION: Dimension

DESCRIPTION:
The DM command defines a single dimensional array with a name and n total elements. The first element of the defined array starts with element number 0 and the last element is at n-1.

ARGUMENTS: DM c[n] where
c is a array name of up to eight alphanumeric characters, starting with an uppercase alphabetic character.
n is the number of array elements from 1 to 400.
DM? Returns the number of array elements available.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_DM contains the available array space. On the controller, before any arrays have been defined, the operand _DM will return 400. If an array of 100 elements is defined, the operand _DM will return 300.

RELATED COMMANDS:
"DA" Deallocate Array

EXAMPLES:
```
DM Pets[5],Dogs[2],Cats[3]         Define dimension of arrays, pets with 5
elements; Dogs with 2 elements; Cats with 3 elements
DM Tests[200]          Define dimension of array Tests with 200 elements
```

DR

FUNCTION: Configures I/O Data Record Update Rate

DESCRIPTION:
The controller creates a QR record and sends it periodically to a UDP Ethernet Handle

ARGUMENTS: DR n, m
n specifies the data update rate in msec between updates. n=0 to turn it off, or n must be an integer of at least 8.
m specifies the Ethernet handle on which to periodically send the Data Record. 0 is handle A, 1 is B, 2 is C. The handle must be UDP (not TCP).

Mapping for DataRecord:
header UL
sample number UW
error code UB
general status UB
analog out 0 (counts) UW
analog out 1 (counts) UW
analog out 2 (counts) UW
analog out 3 (counts) UW
analog out 4 (counts) UW
analog out 5 (counts) UW
analog out 6 (counts) UW
analog out 7 (counts) UW
analog in 0 (counts) UW
analog in 1 (counts) UW
analog in 2 (counts) UW
analog in 3 (counts) UW
analog in 4 (counts) UW
analog in 5 (counts) UW
analog in 6 (counts) UW
analog in 7 (counts) UW
output state UW
input state UW
pulse count UL
ZC data SL
ZD data SL

USAGE:
Default Value DR0 (off)
In a Program Yes Default Format --
Command Line Yes

OPERAND USAGE:
_DR contains the data record update rate.

RELATED COMMANDS:
QR Query a single data record

EXAMPLES:
```
:DR1000,0
:G□x□□□□□~□□□P□
_□`□□□□□@~□□□P□
_□H□□□□□`~□□□P□
_□0□□□□□~□□□P□
DR0
```

Note: The data record is in a binary, non-printable format (the output above is normal when printing to the terminal)

ED

FUNCTION: Edit

DESCRIPTION:
Using Telnet style interface (not Galil Software). The ED command puts the RIO board into the Edit subsystem. In the Edit subsystem, programs can be created, changed or destroyed. The commands in the Edit subsystem are:

{cntrl}D	Deletes a line
{cntrl}I	Inserts a line before the current one
{cntrl}P	Displays the previous line
{cntrl}Q	Exits the Edit subsystem
{return}	Saves a line

Using Galil Windows Terminal Software: The ED command causes the Windows terminal software to open the terminal editor.

ARGUMENTS: ED n where
n specifies the line number to begin editing. The default line number is the last line of program space with commands.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:
_ED contains the line number of the last line to have an error

RELATED COMMANDS:
EXAMPLES:

```
000 #BEGIN
001 CB1
002 SB3
003 SLKJ         Bad line
004 EN
005 #CMDERR         Routine which occurs upon a command error
006 V=_ED
007 MG "An error has occurred" {n}
008 MG "In line", V{F3.0}
009 AB
010 ZS0
011 EN
HINT:  Remember to quit the Edit Mode prior to executing or listing a program.
```

ELSE

FUNCTION: Else function for use with IF conditional statement

DESCRIPTION:
The ELSE command is an optional part of an IF conditional statement. The ELSE command must occur after an IF command and it has no arguments. It allows for the execution of a command only when the argument of the IF command evaluates False. If the argument of the IF command evaluates false, the controller will skip commands until the ELSE command. If the argument for the IF command evaluates true, the RIO board will only execute the commands between the IF and ELSE command and then will skip to ENDIF.

ARGUMENTS: ELSE

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"ENDIF" End of IF conditional Statement

EXAMPLES:
```
IF (@IN[1]=0)          IF conditional statement based on input 1
IF (@IN[2]=0)          2nd IF executed if 1st IF is true
MG "INPUT 1 AND INPUT 2 ARE INACTIVE"          Executed if 2nd IF is true
ELSE                   ELSE command for 2nd IF statement
MG "ONLY INPUT 1 IS ACTIVE          Executed if 2nd IF is false
ENDIF          End of 2nd conditional statement
ELSE          ELSE command for 1st IF conditional statement
MG"ONLY INPUT 2 IS ACTIVE"          Executed if 1st IF is false
ENDIF          End of 1st conditional statement
```

EN

FUNCTION: End

DESCRIPTION:
The EN command is used to designate the end of a program or subroutine. If a subroutine was called by the JS command, the EN command ends the subroutine and returns program flow to the point just after the JS command. No argument is required.
The EN command is also used to end the automatic subroutine, #CMDERR, and an argument is required to handle trippoints in the main thread upon returning from the subroutine.

ARGUMENTS: EN m
m=0 Return from #CMDERR without restoring trippoint in main thread
m=1 Return from #CMDERR and restore trippoint in main thread
Note 1: The default value for the argument is 0. For example EN0 and EN have the same effect.
Note 2: Trippoints cause a program to wait for a particular event. The AI command, for example, waits for specific inputs to go high or low. If the #CMDERR subroutine is executed due to an invalid command while the program is waiting for a trippoint, the #CMDERR can end by continuing to wait for the trippoint as if nothing happened, or clear the trippoint and continue executing the program at the command just after the trippoint. The EN arguments will specify how the #CMDERR routine handles trippoints.
Note 3: Use the RI command to return from the #ININT subroutine.
Note 4: Trippoints occurring in other threads other than the main thread will not be affected by the #CMDERR subroutine.

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"RI"
Return from interrupt subroutine

EXAMPLES:
```
#A          Program A
SB1          Set output 1 high
WT500          Wait for 500 msec
CB1          Set output 1 low
MG "DONE"          Print message
EN          End of Program
```

ENDIF

FUNCTION: End of IF conditional statement

DESCRIPTION:
The ENDIF command is used to designate the end of an IF conditional statement. An IF conditional statement is formed by the combination of an IF and ENDIF command. An ENDIF command must always be executed for every IF command that has been executed. It is recommended that the user not include jump commands inside IF conditional statements, since this causes re-direction of command execution. In this case, the command interpreter may not execute an ENDIF command.

ARGUMENTS: ENDIF

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"ELSE" Optional command to be used only after IF command
"JP" Jump command
"JS" Jump to subroutine command

EXAMPLES:
```
IF (@IN[1]=0)                  IF conditional statement based on input 1
MG "INPUT 1 IS ACTIVE"         Message to be executed if "IF" conditional is
true
ENDIF                           End of conditional statement
```

EO

FUNCTION: Echo

DESCRIPTION:
The EO command turns the echo on or off for the serial port. If the echo is off, characters input over the serial connection will not be echoed back.

ARGUMENTS: EO n where
n=0 or 1. 0 turns echo off, 1 turns echo on.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
_EO contains the state of the echo; 0 is off, 1 is on

RELATED COMMANDS:
EXAMPLES:

```
EO 0          Turns echo off
EO 1          Turns echo on
```

@FRAC[n]

FUNCTION: Fractional part

DESCRIPTION:
Returns the fractional part of the given number

ARGUMENTS: @FRAC[n]
n is a signed number in the range -2147483648 to 2147483647.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@INT[n]
Integer part

EXAMPLES:

```
:MG @FRAC[1.2]
0.2000
:MG @FRAC[-2.4]
-0.4000
```

HS

FUNCTION: Handle Assignment Switch

DESCRIPTION:
The HS command is used to switch the handle assignments between two handles. Handles are opened on the RIO when a connection is established by a PC, or a handle is assigned explicitly with the IH command. Should those assignments need modifications, the HS command allows the handles to be reassigned.

ARGUMENTS: HSh=i where
h is the first handle of the switch (A,B or C)
i is the second handle of the switch (A,B or C)

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"IH"
Internet Handle

EXAMPLES:
```
HSA=B          Connection for handle A is assigned to handle B. Connection for
handle B is assigned to handle A.
```

HX

FUNCTION: Halt Execution

DESCRIPTION:
The HX command halts the execution of any of the 4 programs that may be running independently in multitasking. The parameter n specifies the thread to be halted.

ARGUMENTS: HX n where
n is 0 to 3 to indicate the 4 threads

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
When used as an operand, _HX n contains the running status of thread n with:

0	Thread not running
1	Thread is running
2	Thread has paused at trippoint

RELATED COMMANDS:
"XQ"
Execute program

EXAMPLES:

```
XQ #A          Execute program #A, thread zero
XQ #B,1          Execute program #B, thread one
HX0          Halt thread zero
HX1          Halt thread one
```

IA

FUNCTION: IP Address

DESCRIPTION:
The IA command assigns the controller with an IP address.
The IA command may also be used to specify the time out value. This is only applicable when using the TCP/IP protocol.

ARGUMENTS: IA ip0,ip1,ip2, ip3 or IA n or IA
ip0, ip1, ip2, ip3 are 1 byte numbers separated by commas and represent the individual fields of the IP address.
n is the IP address for the controller, which is specified as an integer representing the signed 32 bit number (two's complement).
> u specifies the multicast IP address where u is an integer between 0 and 63.
< t specifies the time in update samples between TCP retries.
IA? will return the IP address of the RIO board.

USAGE:
In a Program No
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_IA0 contains the IP address representing a 32 bit signed number (Two's complement)
_IA1 contains the value for t (retry time)
_IA2 contains the number of available handles
_IA3 contains the number of the handle using this operand where the number is 0 to 2. 0 represents handle A, 1 for handle B, 2 for C.
_IA4 contains the handle that lost communication last. Contains -1 on reset to indicate no handles lost.
_IA5 contains the Ethernet speed, 10 or 100 Mbits/sec

RELATED COMMANDS:
"IH"
Internet Handle

EXAMPLES:
```
IA 151, 12, 53, 89         Assigns the controller with the address 151.12.53.89
IA 2534159705          Assigns the controller with the address 151.12.53.89
IA < 500          Sets the timeout value to 500msec
```

ID

FUNCTION: Identify

DESCRIPTION:
The ID command is used to query the RIO to identify the I/O hardware available.

ARGUMENTS: None

USAGE:
Default Value -----
In a Program No Default Format -----
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
:ID
outputs 0-3 = power sourcing outputs
outputs 4-7 = power sourcing outputs
outputs 8-11 = sinking outputs
outputs 12-15 = sinking outputs
```

IF

FUNCTION: IF conditional statement

DESCRIPTION:
The IF command is used in conjunction with an ENDIF command to form an IF conditional statement. The arguments are one or more conditional statements. If the conditional statement(s) evaluates true, the command interpreter will continue executing commands which follow the IF command. If the conditional statement evaluates false, the controller will ignore commands until the associated ENDIF command OR an ELSE command occurs in the program.

ARGUMENTS: IF condition where
Conditions are tested with the following logical operators:
< less than or equal to
> greater than
= equal to
<= less than or equal to
>= greater than or equal to
<> not equal

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"ELSE" Optional command to be used only after IF command
"ENDIF" End of IF conditional Statement

EXAMPLES:
```
IF (@IN[1]=0)                  IF conditional statement based on input 1
MG "Input 1 is Low"          Message to be executed if "IF" statement is true
ENDIF                            End of IF conditional statement
  IF(V1>25)&(@IN[4]=1)         Conditions based on V1 variable and input 4
status
MG "Conditions met"          Message to be executed if "IF" statement is true
ENDIF                            End of IF statement
```

IH

FUNCTION: Open Internet Handle

DESCRIPTION:
The IH command is used when the controller is operated as a master (also known as a client). This command opens a handle and connects to a slave (server).
Each RIO board may have 3 handles open at any given time. They are designated by the letters A, B and C. To open a handle, the user must specify:
1. The IP address of the slave
2. The type of session: TCP/IP or UDP/IP
3. The port number of the slave. This number is not necessary if the slave device does not require a specific port value. If not specified, the board will specify the port value as 1000.

ARGUMENTS: IHh= ip0,ip1,ip2,ip3

q or IHh=n

q or IHh= >r where
h is the handle, specified as A, B, or C
ip0,ip1,ip2,ip3 are integers between 0 and 255 and represent the individual fields of the IP address. These values must be separated by commas.
n is a signed integer between - 2147483648 and 2147483647. This value is the 32 bit IP address and can be used instead of specifying the 4 address fields.
IHS => C closes the handle that sent the command; where C=-1 for UDP/IP, or C=-2 for TCP/IP.
IHN => C closes all handles except for the one sending the command; where C=-1 UDP, or C=-2 TCP.

>q specifies the connection type where q is 0 for no connection, 1 for UDP and 2 for TCP
>r specifies that the connection be terminated and the handle be freed, where r is -1 for UDP, -2 for TCP/IP or -3 for TCP/IP reset
"?" returns the IP address as 4 1-byte numbers

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_IHh0 contains the IP address as a 32 bit number

_IHh1 contains the slave port number

_IHh2 contains a 0 if the handle is free
contains a 1 if it is for a UDP slave
contains a 2 if it is for a TCP slave
contains a -1 if it is for a UDP master
contains a -2 if it is for a TCP master
contains a -5 while establishing a UDP handle
contains a -6 while establishing a TCP handle

_IHh3 contains a 0 if the ARP was successful
contains a 1 if it has failed or is still in progress.

_IHh4 contains a 1 if the master controller is waiting for acknowledgment from the slave after issuing a command.
contains a 2 if the master controller received a colon from the slave after issuing a command.
contains a 3 if the master controller received a question mark from the slave after issuing a

command.
contains a 4 if the master controller timed-out while waiting for a response from the slave after issuing a command.

RELATED COMMANDS:
"IA" Internet Address

EXAMPLES:
```
IHA=251,29,51,1          Open handle A at IP address 251.29.51.1
IHA= -2095238399          Open handle A at IP address 251.29.51.1
Note:  When the IH command is given, the controller initializes an ARP on the
slave device before opening a handle.  This operation can cause a small time
delay before the RIO board responds.
```

II

FUNCTION: Input Interrupt

DESCRIPTION:
This newly formatted II command enables the interrupt function for the specified inputs. This function can trigger one of four input interrupt subroutines (#ININT) when the controller sees that the conditional statement is satisfied.
When the condition is satisfied, the program will jump to the subroutine with label #ININTn, where n ranges from 0 to 3. This subroutine will be executed in thread m, where m is between 0 (main) and 3, causing any trippoint set in that thread to be cleared. But this trippoint can be re-enabled by the proper termination of the interrupt subroutine using RI. The RI command is used to return from the #ININTn routines.
To avoid returning to the program on an interrupt, use the command ZS to zero the subroutine stack and use the II command to reset the interrupt.
Note: An application program must be running on the controller for the interrupt function to work.

ARGUMENTS: II n,m,condition where
n is an integer between 0 and 3. This number specifies the #ININTn subroutine to be executed when the interrupt occurs.
m is an integer between 0 and 3. This argument indicates the thread number in which the #ININTn subroutine is going to be executed. The specified thread needs to be running when the interrupt occurs, otherwise the #ININTn subroutine will not be executed. Upon interrupt, the existing thread m will be interrupted to allow the execution of the interrupt subroutine. Upon completion of the interrupt, the main program in thread m will once again be enabled from the point at which the interrupt occurred.
Condition can consist of any number of inputs, using the "&" operator between each input. A positive input number means the condition is for that input to go high, and a negative input number for that input to go low.

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"RI" Return from Interrupt
#ININTn Interrupt Subroutine
"AI" Trippoint for input

EXAMPLES:
```
#A                          Program A
II 2,0,3&5&-10                 Specify interrupt #2 on main thread when
                            inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP                 Loop
EN                          End Program
#ININT2                          Interrupt subroutine number 2
MG "INTERRUPT"                 Print Message
#CLEAR;
JP#CLEAR,@IN[1]=0         Check for 'reset' input 1 to clear interrupt
RI                          Return to main program
```

IL

FUNCTION: Integrator Limit

DESCRIPTION:
The IL command limits the effect of the integrator gain in the process control loop to a certain voltage. For example, IL 2 limits the output of the integrator of the A-channel to the +/-2 Volt range. Note: the output from the KD and KP terms is not affected.
For further details see the Process Control section in the User's Manual.

ARGUMENTS: IL n,n where
n is a number in the range 0 to 5 Volts with a resolution of 0.0001.
n = ? Returns the value of the integrator limit

USAGE:
Default Value 5
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
_ILn contains the value of the integrator limit

RELATED COMMANDS:
KI Integrator

EXAMPLES:
```
KI 2,4                  Integrator constants
IL 3,2                  Integrator limits
IL ?                  Returns the first limit
3.0000
```

IN

FUNCTION: Input Variable

DESCRIPTION:
The IN command allows a variable to be input from a keyboard. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.
The IN command holds up execution of following commands in a program until a carriage return or semicolon is detected. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts will still be active.

ARGUMENTS: IN "m" , n {So} where
"m" is the prompt message. May be letters, numbers, or symbols up to maximum line length and must be placed in quotations. Make sure that maximum line length of 40 characters is not exceeded.
n is the name of variable to hold value returned from input.
{So} specifies string data where o is the number of characters from 1 to 6. Not required if the data is numerical.
Note 1: Do not leave a space between the comma and n.
Note2: The IN command cannot be used over Ethernet, only RS-232.
Note 3: IN command can only be used in thread 0.

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES: Operator specifies a bit that is to be turned on.

```
#A
'Prompt operator for bit to set high
IN "Enter bit number to turn on",N1
SBN1;    'Set the specified bit high
MG "DONE";   'Print Message
EN;          'End Program
```

@IN[n]

FUNCTION: Read digital input

DESCRIPTION:
Returns the value of the given digital input (either 0 or 1)

ARGUMENTS: @IN[n] where
n is an unsigned integer in the range 0 to 15

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@AN[n] Read analog input
@OUT[n] Read digital output
SB Set digital output bit
CB Clear digital output bit
OF Set analog output offset

EXAMPLES:
```
MG @IN[1]          print digital input 1
:1.0000
x = @IN[1]          assign digital input 1 to a variable
x = ?                    query variable
:1.000
```

#ININTn

FUNCTION: Input interrupt automatic subroutine

DESCRIPTION:
The RIO provides an input interrupt function which causes the program to automatically execute the instructions following the #ININTn label, where n ranges from 0 to 3. This function is enabled using the II n,m,condition command, where n specifies the #ININTn subroutine to be executed, m specifies the thread number in which the interrupt subroutine #ININTn is going run, and condition is the input sequence that needs to be true for the #ININTn routine to be executed. For more details, see the II command page.

Note that the specified thread needs to be executing at the time of the interrupt otherwise the #ININTn routine will be ignored

ARGUMENTS: #ININTn where
n is an integer in the range of 0 to 3

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"RI" Return from Interrupt
"II" Interrupt Interrupt

EXAMPLES:

```
#A                          Program A
II 2,0,3&5&-10                 Specify interrupt #2 on main thread when
                            inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP                 Loop
EN                          End Program
#ININT2                          Interrupt subroutine number 2
MG "INTERRUPT"                 Print Message
#CLEAR;
JP#CLEAR,@IN[1]=0         Check for 'reset' input 1 to clear interrupt
RI                          Return to main program
```

@INT[n]

FUNCTION: Integer part

DESCRIPTION:
Returns the integer part of the given number.

ARGUMENTS: @INT[n]
n is a signed number in the range -2147483648 to 2147483647.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@FRAC[n] Fractional part

EXAMPLES:

```
:MG @INT[1.2]
1.0000
:MG @INT[-2.4]
-2.0000
```

IQ

FUNCTION: Digital Input Configuration

DESCRIPTION:
The IQ command sets the active level for each of the 16 digital inputs. A 0 means current flowing through the opto is logic 1, a 1 means current *not* flowing through the opto is logic 1. Current flowing through the opto can be seen on the RIO when the green LED turns ON. The default state is 0 which means that current not flowing through the opto (LED off) is logic 1.

ARGUMENTS: IQn where
n is an integer value 0-65535 representing the 16 digital inputs

USAGE:
Default Value 0
In a Program Yes Default Format -----
Command Line Yes

OPERAND USAGE:
_IQn contains the current setting

RELATED COMMANDS:
EXAMPLES:

```
:IQ255;'This sets bank0 inputs to show current not flowing is logic 0
:IQ192;'This sets outputs 6 and 7 to show current not flowing is logic 0
```

JP

FUNCTION: Jump to Program Location

DESCRIPTION:
The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true. Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions requires that only one statement be true for the combined statement to be true.
Note 1: Each condition must be placed in parenthesis for proper evaluation by the controller.

ARGUMENTS: JP location,condition where
location is a program line number or label
condition is an optional conditional statement using a logical operator
The logical operators are:

- < less than
- > greater than
- = equal to
- <= less than or equal to
- >= greater than or equal to
- <> not equal to

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
JP #POS1,V1<5          Jump to label #POS1 if variable V1 is less than 5
JP #A,@IN[1]=0          Jump to #A if input 1 is low
JP #B          Jump to #B (no condition)
HINT:  JP is similar to an IF, THEN command.  Text to the right of the comma is
the condition that must be met for a jump to occur.  The destination is the
specified label before the comma.
```

JS

FUNCTION: Jump to Subroutine

DESCRIPTION:
The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine.
Multiple conditions can be used in a single jump subroutine statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions requires that only one statement be true for the combined statement to be true.
Note: Each condition must be placed in parenthesis for proper evaluation by the controller.
Note: Subroutines can be nested 16 deep in the controller.

ARGUMENTS: JS destination,condition where
destination is a line number or label
condition is an optional conditional statement using a logical operator
The logical operators are:

- < less than or equal to
- > greater than
- = equal to
- <= less than or equal to
- >= greater than or equal to
- <> not equal

USAGE:
In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"EN"
End

EXAMPLES:

```
JS #B,N1<5          Jump to subroutine #B if N1 is less than 5
JS #LOOP,N1<>0          Jump to #LOOP if N1 is not equal to 0
JS #A          Jump to subroutine #A (no condition)
```

KD

FUNCTION: Derivative Constant

DESCRIPTION:
KD designates the derivative constant in the control filter. The filter transfer function is

D(z) = KP + KD(z-1)/z + KIz/2 (z-1)

The derivative gain outputs a voltage based on the rate of change of the error. For further details see the Process Control section in the User's Manual.

ARGUMENTS: KD n,n where
n is an unsigned numbers in the range 0 to 4095 with a resolution of 1/8.
n = ? Returns the value of the derivative constant for the specified channel.

USAGE:
Default Value 64
In a Program Yes Default Format 4.2
Command Line Yes

OPERAND USAGE:
_KDn contains the value of the derivative constant for the specified channel.

RELATED COMMANDS:
KI Integrator
KP Proportional

EXAMPLES:
```
KD 100,200          Specify KD
KD ?,?                   Return KD
:100.00, 200.00
```

KI

FUNCTION: Integrator

DESCRIPTION:
The KI command sets the integral gain of the control loop. It fits in the control equation as follows:

D(z) = KP + KD(z-1)/z + KI z/2(z-1)

The integrator term will reduce the error at rest to zero. For further details see the Process Control section in the User's Manual.

ARGUMENTS: KI n,n or KIx=n where
n is an unsigned numbers in the range 0 to 255 with a resolution of 0.001.
n = ? Returns the value for the specified channel.

USAGE:
Default Value 0
In a Program Yes Default Format 4.4
Command Line Yes

OPERAND USAGE:
_KIn contains the value of the integral gain for the specified channel.

RELATED COMMANDS:
KP Proportional Constant
KD Derivative Constant
IL Integrator Limit

EXAMPLES:

```
KI 12,14          Specify A,B channel integral gain
KI 7                   Specify A channel only
KI ,8                   Specify B channel only
KI ?,?                   Return A,B
:7, 14                   KI values
```

KP

FUNCTION: Proportional Constant

DESCRIPTION:
KP designates the proportional constant in the controller filter. The filter transfer function is
D(z) = KP + KD(z-1)/z + KI z/2(z-1)
The proportianal gain outputs a voltage proportional to the amount of error. For further details see the Process Control section in the User's Manual.

ARGUMENTS: KP n,n or KPA=n where
n is an unsigned numbers in the range 0 to 1023.875 with a resolution of 1/8.
n = ? Returns the value of the proportional constant for the specified channel.

USAGE:
Default Value 6
In a Program Yes Default Format 4.2
Command Line Yes

OPERAND USAGE:
_KPn contains the value of the proportional constant for the specified channel.

RELATED COMMANDS:
KD Derivative Constant
KI Integrator Constant
IL Integrator Limit

EXAMPLES:
```
KP 12,14          Specify A,B channel proportional gain
KP 7                   Specify A channel only
KP ,8                   Specify B channel only
KP ?,?                   Return A,B channel
:7, 14                   KP values
```

LA

FUNCTION: List Arrays

DESCRIPTION:
The LA command returns a list of all arrays in memory. The listing will be in alphabetical order. The size of each array will be included next to each array name in square brackets.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"LL" List Labels
"LS" List Program
"LV" List Variable

EXAMPLES:
```
: LA          Interrogation Command
CA [10]          RIO board returns a list of 4 Arrays
LA [5]
NY [25]
VA [17]
```

{control}L{control}K

FUNCTION: Lock program

DESCRIPTION:
{control}L{control}K locks user access to the application program. When locked, the ED, UL, LS, and TR commands will give privilege error #106. The application program will still run when locked.
The locked or unlocked state can be saved with a BN command. Upon master reset, the controller is unlocked. Once the program is unlocked, it will remain accessible until a lock command or a reset (with the locked condition burned in) occurs.

ARGUMENTS: {control}L{control}Kpassword,n where
When n is 1, this command will lock the application program.
When n is 0, the program will be unlocked.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
PW Password
ED Edit program
UL Upload program
LS List program
TR Trace program

EXAMPLES:
```
:PW test,test         Set password to "test"
:^L^Ktest,1         Lock the program
:ED         Attempt to edit program
```

LL

FUNCTION: List Labels

DESCRIPTION:
The LL command returns a listing of all of the program labels in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"LV"
List Variables

EXAMPLES:
```
: LL          Interrogation Command
# FIVE          RIO board returns a list of 5 labels
# FOUR
# ONE
# THREE
# TWO
```

LS

FUNCTION: List

DESCRIPTION:
The LS command sends a listing of the program memory. The listing will start with the line pointed to by the first parameter, which can be either a line number or a label. If no parameter is specified, it will start with line 0. The listing will end with the line pointed to by the second parameter--again either a line number or label. If no parameter is specified, the listing will go to the last line of the program.

ARGUMENTS: LS n,m where
n,m are valid numbers from 0 to 199, or labels. n is the first line to be listed, m is the last.

USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
:LS #A,6          List program starting at #A through line 6
002 #A
003 MG "Program A"
004 COUNT=1
005 SB2
006 WT 2000
```

LV

FUNCTION: List Variables

DESCRIPTION:
The LV command returns a listing of all of the program variables in memory (max number of variables is 126). The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"LL"
List Labels

EXAMPLES:

```
: LV          Interrogation Command
APPLE  = 60.0000          RIO board returns a list of 3 variables
BOY = 25.0000
ZEBRA = 37.0000
```

LZ

FUNCTION: Inhibit leading zeros

DESCRIPTION:
The LZ command is used for formatting the values returned from interrogation commands or interrogation of variables and arrays. By enabling the LZ function, all leading zeros of returned values will be removed.

ARGUMENTS: LZ n where n is
1 to remove leading zeros
0 to disabled the leading zero removal
LZ? Returns the state of the LZ function.
Note: Default value is 1.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_LZ contains the state of the LZ command

RELATED COMMANDS:
EXAMPLES:

```
TB          Tell status
:001
LZ 1          Inhibit leading zeros
TB          Tell status
:1
```

MA

FUNCTION: Email Server IP Address

DESCRIPTION:
The MA command sets the SMTP email server ip address. The default address is 0.0.0.0 which yields no email transactions.

ARGUMENTS: MA n,n,n,n where
n is a value from 0 to 254 for each field of the IP address

USAGE:
Default Value 0,0,0,0
In a Program Yes Default Format 3.0
Command Line Yes

OPERAND USAGE:
_MA

RELATED COMMANDS:
"MD" Set Mail Destination Address
"MS" Set Mail Source Address

EXAMPLES:

MB

FUNCTION: Modbus

DESCRIPTION:
The MB command is used to communicate with I/O devices using the first two levels of the Modbus protocol.
The format of the command varies depending on each function code. The function code, -1, designates that the first level of Modbus is used (creates raw packets and receives raw data). The other codes are the 10 major function codes of the second level that the RIO board supports.

Function Code	Definition
01	Read Coil Status (Read Bits)
02	Read Input Status (Read Bits)
03	Read Holding Registers (Read Words)
04	Read Input Registers (Read Words)
05	Force Single Coil (Write One Bit)
06	Preset Single Register (Write One Word)
07	Read Exception Status (Read Error Code)
15	Force Multiple Coils (Write Multiple Bits)
16	Preset Multiple Registers (Write Words)
17	Report Slave ID

Note: For those command formats that have "addr", this is the slave address. The slave address is usually zero.
Note: All the formats contain an h parameter. This designates the connection handle (A, B or C).

ARGUMENTS:
MBh = -1, len, array[] where
len is the number of the bytes
Array[] is the name of array containing data
MBh = addr, 1, m, n, array[] where
m is the starting bit number
n is the number of bits
array[] of which the first element will hold result
MBh = addr, 2, m, n, array[] where
m is the starting bit number
n is the number of bits
array[] of which the first element will hold result
MBh = addr, 3, m, n, array[] where
m is the starting register number
n is the number of registers
array[] will hold the response
MBh = addr, 4, m, n, array[] where
m is the starting register number
n is the number of registers
array[] will hold the response
MBh = addr, 5, m, n where
m is the starting bit number
n is 0 or 1 and represents the coil set to off or on.
MBh = addr, 6, m, n where
m is the register number
n is the 16 bit value
MBh = addr, 7, array[] where
array[] is where the returned data is stored (one byte per element)
MBh = addr, 15, m, n, array[] where
m is the starting bit number
n is the number of bits
array[] contains the data (one byte per element)
MBh = addr, 16, m, n, array[] where
m is the starting register number

n is the number of registers
array[] contains the data (one 16 bit word per element)
MBh = addr, 17, array[] where
array[] is where the returned data is stored

USAGE:
In a Program Yes
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

MD

FUNCTION: Email Destination Address

DESCRIPTION:
The MD command sets the email destination address (max 30 characters) - default is null which yields no email transactions.

ARGUMENTS: MD string where
string is the email address to send to

USAGE:
Default Value null
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
_MD

RELATED COMMANDS:
"MA" Set Email Server IP Address
"MS" Set Mail Source Address

EXAMPLES:

MG

FUNCTION: Message

DESCRIPTION:
The MG command transmits data from the RIO board. This can be used to alert an operator, send instructions or return a variable value.

ARGUMENTS: MG "m", {Ex}, {^n}, V {Fm.n or $m,n} {N} {Sn} {M}
"m" is a text message including letters, numbers, symbols or G. Make sure that maximum line length of 40 characters is not exceeded.
{Ex}for ethernet and 'x' specifies the ethernet handle (A, B or C) for sending the message through. {Ex} can be replaced by {P1} to send the message through the serial port. These settings are only needed when the user wants to specify another return port than the one set by the CF command.
{^n} is an ASCII character specified by the value n
V is a variable name or array element, where the following specifiers can be used for formatting:
{Fm.n} Display variable in decimal format with m digits to left of decimal, and n to the right.
{$m,n} Display variable in hexadecimal format with m digits to left of decimal, and n to the right.
{N} Suppress carriage return line feed.
{Sn} Display variable as a string of length n where n is 1 thru 6
{M} emails the message out through an SMTP email server (see MA, MS, and MD commands)
Note: Multiple text, variables, and ASCII characters may be used, each must be separated by a comma. Formatting fields are optional.
Note: The order of arguments is not important.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
Case 1:  Message command displays ASCII strings
         MG "Good Morning"          Displays the string
Case 2:  Message command displays variables or arrays
         MG "The Answer is", Total {F4.2}  Displays the string with the content
of variable TOTAL in local format of 4 digits before and 2 digits after the
decimal point.
Case 3:  Message command sends any ASCII characters to the port.
         MG {^13}, {^30}, {^37}, {N}  Sends carriage return, characters 0 and 7
followed by no carriage return line feed command to the port.
```

MS

FUNCTION: Email Source Address

DESCRIPTION:
The MS command sets the email source address (max 30 characters) - default is RIO@galilmc.com

ARGUMENTS: MS string where
string is the email source address

USAGE:
Default Value null
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
_MS

RELATED COMMANDS:
"MA" Set Email Server IP Address
"MD" Set Mail Destination Address

EXAMPLES:

```
MA 10,0,0,1;                    example smtp email server ip
MD someone@example.com                sample email address
MG "Testing Email"{M}                Message to send via Email
```

MW

FUNCTION: Modbus Wait

DESCRIPTION:
Enabling the MW command causes the controller to hold up execution of the program after sending a Modbus command until a response from the Modbus device has been received. If the response is never received, then the #TCPERR subroutine will be triggered and an error code of 123 will show up in _TC.

ARGUMENTS: MWn where
n = 0 Disables the Modbus Wait function
n = 1 Enables the Modbus Wait function (default)

USAGE:
Default Value 1
In a Program Yes Default Format 1.0
Command Line Yes

OPERAND USAGE:
MW? contains the state of the Modbus Wait.

RELATED COMMANDS:
"MB" Modbus

EXAMPLES:
```
MW1          Enables Modbus Wait
SB1001          Set Bit 1 on Modbus Handle A
CB1001          Clear Bit 1 on Modbus Handle A
Hint:  The MW command ensures that the command that was sent to the Modbus
device was successfully received before continuing program execution.  This
prevents the controller from sending multiple commands to the same Modbus device
before it has a chance to execute them
```

NO (' apostrophe also accepted)

FUNCTION: No Operation

DESCRIPTION:
The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, characters can be given to form a program comment up to the maximum line length of the controller, which is 40. This helps to document a program.
An apostrophe ' may also be used instead of the NO to document a program.

ARGUMENTS: NO m where
m is any group of letters and numbers

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
#A                          Program A
NO                          No Operation
NO This Program                 No Operation
' Does Absolutely         No Operation
' Nothing                  No Operation
EN         End of Program
```

OB

FUNCTION: Output Bit

DESCRIPTION:
The OB n, logical expression command defines output bit n = 0 through 15 as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output.

ARGUMENTS: OB n, expression where
n is 0 through 15, denoting output bit
expression is any valid logical expression, variable or array element.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No
RELATED COMMAND:
"CB" Clear Bit
"SB" Set Bit

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
OB 1, POS=1          If POS 1 is non-zero, Bit 1 is high.
          If POS 1 is zero, Bit 1 is low
OB 2, @IN[5]&@IN[6]          If Input 5 and Input 6 are both high, then
          Output 2 is set high
OB 3, COUNT[1]          If the element 1 in the array is zero, clear bit 3,
otherwise set bit 3
OB N, COUNT[1]          If element 1 in the array is zero, clear bit N
```

OF

FUNCTION: Offset

DESCRIPTION:
The OF command sets a bias voltage on the control output.

ARGUMENTS: OF n,n where
n is a signed number in the range -2.5V to +2.5V with resolution of 0.0001.
n = ? Returns the offset for the specified channel.

USAGE:
Default Value 0
In a Program Yes Default Format 1.4
Command Line Yes

OPERAND USAGE:
_OF contains the offset value

RELATED COMMANDS:
EXAMPLES:

```
OF 1,2          Set control channel A to 1, Control channel B offset to 2
OF -2          Set control channel A to -2  Leave other channel unchanged
OF ,0          Set control channel B to 0  Leave other channel unchanged
OF ?,?          Return offsets
:-2.0000,0.0000
OF ?          Return A offset
:-2.0000
OF ,?          Return B offset
:0.0000
```

OP

FUNCTION: Output Port

DESCRIPTION:
The OP command sends data to the output ports of the controller. You can use the output port to control external switches and relays. The arguments of the OP command are the decimal representation of the general output bits 0 through 7. For example, OP255,0 sets all outputs in bank 0 (bits 0-7) high and all outputs in bank 1 (bits 8-15) low.

ARGUMENTS: OP m,n where
m and n are integers in the range 0 to 255 decimal, or $00 to $FF hexadecimal.

Arguments	Blocks	Bits	Description
m	0	0-7	General Outputs (bank 0)
n	1	8-15	General Outputs (bank 1)

m or n = ? returns the value of the argument

USAGE:

Default Value	0		
In a Program	Yes	Default Format	5.0
Command Line	Yes		

OPERAND USAGE:
_OP0 contains the value of the first argument, m
_OP1 contains the value of the first argument, n

RELATED COMMANDS:

SB	Set output bit
CB	Clear output bit

EXAMPLES:

```
OP 0          Clear Output Port -- all bits
OP $85          Set outputs 1,3,8; clear the others
MG _OP0          Returns the first parameter "m"
MG _OP1          Returns the second parameter "n"
```

@OUT[n]

FUNCTION: Read digital output

DESCRIPTION:
Returns the value of the given digital output (either 0 or 1)

ARGUMENTS: @OUT[n] where
n is an unsigned integer in the range 0 to 15

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@AN[n] Read analog input
@IN[n] Read digital input
SB Set digital output bit
CB Clear digital output bit
OF Set analog output offset

EXAMPLES:
```
MG @OUT[1] ;'print digital output 1
:1.0000
x = @OUT[1] ;'assign digital output 1 to a variable
```

PC

FUNCTION: Pulse Counter Enable

DESCRIPTION:
Enables the use of a pulse counter on input DI3. When turned on, input DI3 will not be available as a general purpose input accessible with IF or @IN[].

ARGUMENTS: PCn where
- n=0 (default) input DI3 is a general purpose input
- n=1 sets input DI3 to be a rising edge pulse counter (also clears the pulse counter)
- n=-1 sets input DI3 to be a falling edge pulse counter (also clears the pulse counter)
- n=? returns the status of the pulse counter (0 if disabled, 1 if enabled)

USAGE:
Default Value 0
In a Program Yes
Command Line Yes

OPERAND USAGE:
_PC contains the number of pulses counted

RELATED COMMANDS:
EXAMPLES:
```
PC1
MG_PC
```

PS

FUNCTION: Control Setpoint

DESCRIPTION:
sdf

ARGUMENTS: PS n,n where
where n = 0 (default) to 5.0000V sets the control loop commanded voltage

USAGE:
Default Value 0
In a Program Yes Default Format 1.4
Command Line Yes

OPERAND USAGE:
_PSn

RELATED COMMANDS:
EXAMPLES:
```
PS
```

PW

FUNCTION: Password

DESCRIPTION:
The password can be set with the command PW password,password where the password can be up to 8 alphanumeric characters. The default value after master reset is a null string. The password can only be changed when the controller is in the unlocked state (^L^K). The password is burnable but cannot be interrogated. If you forget the password you must master reset the controller to gain access.

ARGUMENTS: PW n,n where
n is a string from 0 to 8 characters in length

USAGE:
Default Value "" (null string)
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
LK Lock/Unlock
ED Edit program
UL Upload program
LS List program
TR Trace program

EXAMPLES:

```
:PWtest,test                Set password to "test"
:^L^K test,1                Lock the program
:ED                         Attempt to edit program
```

QD

FUNCTION: Download Array

DESCRIPTION:
The QD command transfers array data from the host computer to the RIO.
QD array[],start,end requires that the array name be specified along with the first element of the array and last element of the array. The array elements can be separated by a comma (,) or by . The downloaded array is terminated by a \.

ARGUMENTS: QD array[],start,end where
"array[]" is a valid array name
"start" is the first element of the array (default=0)
"end" is the last element of the array (default=last element)

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"QU" Upload array

EXAMPLES:
```
Using Galil terminal software, the command can be used in the following manner:
1.          Define the array with the DM command
2.          Set the timeout to 0 (this disengages the timeout)
3.          Send the command QD
3a.  Use the send file command to send the data file.
OR
3b.  Enter data manually from the terminal.  End the data entry with the
character ''
4.            Set the timeout back to a positive number
```

QR

FUNCTION: I/O Data Record

DESCRIPTION:
The QR command causes the RIO to return a record of information regarding the I/O status back to the host PC. This status information includes 4 bytes of header information and specific blocks of I/O information. The details of the status information are described in the user manual.

ARGUMENTS: QR
Mapping for DataRecord:

header	UL
sample number	UW
error code	UB
general status	UB
analog out 0 (counts)	UW
analog out 1 (counts)	UW
analog out 2 (counts)	UW
analog out 3 (counts)	UW
analog out 4 (counts)	UW
analog out 5 (counts)	UW
analog out 6 (counts)	UW
analog out 7 (counts)	UW
analog in 0 (counts)	UW
analog in 1 (counts)	UW
analog in 2 (counts)	UW
analog in 3 (counts)	UW
analog in 4 (counts)	UW
analog in 5 (counts)	UW
analog in 6 (counts)	UW
analog in 7 (counts)	UW
output state	UW
input state	UW
pulse count	UL
ZC data	SL
ZD data	SL

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

OPERAND USAGE:
RELATED COMMANDS:
"QZ" Return Data Record information
Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

QU

EXAMPLES:

QU

FUNCTION: Upload Array

DESCRIPTION:
The QU command transfers array data from the RIO to a host computer. QU requires that the array name be specified along with the first element of the array and last element of the array. The uploaded array will be followed by a Z as an end of text marker.

ARGUMENTS: QU array[],start,end,delim where
"array[]" is a valid array name
"start" is the first element of the array (default=0)
"end" is the last element of the array (default=last element)
"delim" specifies the character used to delimit the array elements. If delim is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"QD" Download array

EXAMPLES:

QZ

FUNCTION: Return Data Record information

DESCRIPTION:
The QZ command is an interrogation command that returns information regarding the Data Record. The RIO board's response to this command will be the return of 4 integers separated by commas.

The first field returns the number of control loops

The second field returns the number of bytes in the general data block of the QR record. This is always 4 for the RIO.

Third field returns the number of bytes in the I/O block of the QR record

Fourth field returns 0, representing nothing

ARGUMENTS: QZ

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"QR" Data Record

EXAMPLES:
```
QZ          Enter Command
2, 4, 48, 0
```

RA

FUNCTION: Record Array

DESCRIPTION:
The RA command selects one or two arrays for automatic data capture. The selected arrays must have been dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

ARGUMENTS: RA n [],m [] where
n,m are dimensioned arrays as defined by DM command. The [] contain nothing.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
"DM" Dimension Array
"RD" Record Data
"RC" Record Interval

EXAMPLES:
```
#Record          Label
DM Input[100]          Define array for input status
RA Input[]          Specify Record Mode
RD _TI          Specify data type for record
RC 1          Begin recording at 2 msec intervals
EN          End
```

RC

FUNCTION: Record

DESCRIPTION:
The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording.

ARGUMENTS: RC n,m where
n is an integer 1 thru 8 and specifies 2n msec between records. RC 0 stops recording.
m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used. A negative number for m causes circular recording over array addresses 0 to m-1. The address for the array element for the next recording can be interrogated with _RD.
RC? returns status of recording. '1' if recording, '0' if not recording.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_RC contains status of recording, '1' if recording, '0' if not recording.

RELATED COMMANDS:
"DM" Dimension Array
"RA" Record Array Mode
"RD" Record Data

EXAMPLES:
```
See example for RA command.
```

RD

FUNCTION: Record Data

DESCRIPTION:
The RD command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

DATA TYPE	MEANING
_TI	Input status (0-15)
_OP	Output status (0-15)
_AFn	Analog inputs status (0-7)
_AOn	Analog outputs (0-7)

ARGUMENTS: RD m1, m2 where
the arguments are the data type to be captured using the record array feature. The order is important. Each of the two data types corresponds with the array specified in the RA command.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:
_RD contains the address for the next array element for recording.

RELATED COMMANDS:

"DM"	Dimension Array
"RA"	Record Array
"RC"	Record Interval

EXAMPLES:
```
See example for RA command.
```

RE

FUNCTION: Return from Error Routine

DESCRIPTION:
The RE command is used to end the error handling subroutine #TCPERR. An RE at the end of this routine causes a return to the main program. Care should be taken to be sure the error condition no longer occurs to avoid re-entering the subroutine. If the program sequencer was waiting for a trippoint to occur, prior to the error interrupt, the trippoint condition is preserved on the return to the program if RE1 is used. RE0 clears the trippoint. To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack.

ARGUMENTS: RE n where
n = 0 Clears the interrupted trippoint
n = 1 Restores state of trippoint
no argument clears the interrupted trippoint

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
#TCPERR Error Subroutine

EXAMPLES:

```
#L
  MG {EA} "L"
  WT1000
JP#L
#TCPERR
  MG {P1} "TCPERR.  Dropped handle", _IA4
RE
```

REM

FUNCTION: Remark

DESCRIPTION:
REM is used for comments. The REM statement is NOT a controller command. Rather, it is recognized by Galil PC software, which strips away the REM lines before downloading the DMC file to the controller. REM differs from NO (or ') in the following ways:
(1) NO (or ') comments are downloaded to the controller and REM comments aren't
(2) NO (or ') comments take up execution time and REM comments don't; therefore, REM should be used for code that needs to run fast.
(3) REM comments cannot be recovered when uploading a program but NO (or ') comments are recovered. Thus the uploaded program is less readable with REM.
(4) NO (or ') comments take up program line space and REM lines don't.
(5) REM comments must be the first and only thing on a line, whereas NO (or ') can be used to place comments to the right of code (after a semicolon) on the same line

NO (or ') should be used instead of REM unless speed or program space is an issue.

ARGUMENTS: REM string where
string is a text comment

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
NO (' apostrophe also accepted)
No operation (comment)

EXAMPLES:
```
REM This comment will be stripped when downloaded to the controller
'This comment will be downloaded and takes some execution time
PRX=1000; 'this comment is to the right of the code
```

RI

FUNCTION: Return from Interrupt Routine

DESCRIPTION:
The RI command is used to end the interrupt subroutine beginning with the label #ININTn. If the program sequencer was interrupted while waiting for a trippoint (such as WT), RI1 restores the trippoint on the return to the program. RI0 clears the trippoint. The second field of the RI command either restores or disables the input interrupt feature. This field has been added as a special feature of the RIO. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

ARGUMENTS: RI m,n where
m = 0 or 1
0 clears trippoint (e.g. AI trippoint)
1 restores trippoint
n = 0 (or no field) or 1
0 restores interrupt
1 disables interrupt

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
#ININTn Input interrupt subroutine
"II"
Enable input interrupts

EXAMPLES:
```
#A;II1,0,3          Program label; enable interrupt on input 3
AI5;MG"DONE";EN          AI trippoint on input 5; end of program
#ININT1          Begin interrupt subroutine
MG "IN[3] INTERRUPTED"          Print Message
CB 3          Set output line 1
RI 1,1          Return to the main program, restore AI trippoint and disable
interrupt on input 3
HINT:  An applications program must be executing for the #ININTn subroutine to
function.
```

@RND[n]

FUNCTION: Round

DESCRIPTION:
Rounds the given number to the nearest integer

ARGUMENTS: @RND[n]
n is a signed number in the range -2147483648 to 2147483647.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@INT[n] Truncates to the nearest integer

EXAMPLES:
```
:MG @RND[1.2]
1.0000
:MG @RND[5.7]
6.0000
:MG @RND[-1.2]
-1.0000
:MG @RND[-5.7]
-6.0000
:MG @RND[5.5]
6.0000
:MG @RND[-5.5]
-5.0000
```

RS

FUNCTION: Reset

DESCRIPTION:
The RS command resets the state of the processor to its power-on condition. The previously saved state of the RIO, along with parameter values and saved sequences, are restored.

ARGUMENTS:
USAGE:
In a Program No
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_RS returns the state of the processor on its last power-up condition. The value returned is the decimal equivalent of the 4 bit binary value shown below.

Bit 3 For master reset error (there should be no program to execute)
Bit 2 For program check sum error
Bit 1 For parameter check sum error
Bit 0 For variable check sum error

RELATED COMMANDS:
EXAMPLES:

```
RS          Reset the RIO board
```

{control}R{control}S

FUNCTION: Master Reset

DESCRIPTION:
The Master Reset command resets the RIO to factory default settings and erases EEPROM. A master reset can also be performed by installing a jumper at the location labeled MRST and resetting the board (power cycle or pressing the reset button). Remove the jumper after this procedure.

ARGUMENTS:
USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

{control}R{control}V

FUNCTION: Revision Information

DESCRIPTION:
The Revision Information command causes the RIO board to return the firmware revision information.

ARGUMENTS:
USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

SA

FUNCTION: Send Command

DESCRIPTION:
SA sends a command from the RIO to another Galil Ethernet controller (or another RIO unit). Any command can be sent to a slave device and will be interpreted by the controller as a "local" command. Note: A wait statement (e.g. WT5) must be inserted between successive calls to SA.

ARGUMENTS: SAh= arg or SAh=arg,arg,arg,arg,arg,arg,arg,arg where
h is the handle being used to send commands to another Galil Ethernet controller (A, B, or C)
arg is a number, a controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^31)followed by two bytes of fraction (+/- 2,147,483,647.9999). The maximum number of characters for a string is 6 characters. Strings are identified by quotations.
Typical usage would have the first argument as a string such as "OQ" and the subsequent arguments as the arguments to the command: Example SAA= "KD",20,40 would send the command "KD 20,40".

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
_SAhn gives the value of the response to the command sent with an SA command. The h value represents the handle A, B or C and the n value represents the specific field returned from typically a multi-channel controller. The n value gets defaulted to 1 if unspecified in the operand. If a specific field is not used by the controller (e.g. fifth field on a four axes controller), the operand will be -2^31. Response from another RIO board has usually just one field; therefore the n value does not need to be included in the operand.

RELATED COMMANDS:
"IH" Set Internet Handles

EXAMPLES:
```
SAA="KI",1,2          Sends the command to handle A (e.g. a controller):  KI 1,2
WT5
SAA="TE"          Sends the command to handle A (controller):  TE
MG _SAA                   Display the content of the operand _SAA (first response
to TE command)
: 132
MG _SAA2          Display the content of the operand _SAA (2nd response to TE
command)
: 12
Note:  The SA command does not wait for a response from the slave controller
before continuing code execution.  Therefore, a WTxx is required between two SA
commands or between an SA command and querying the response using _SAhn.  There
is a 38 character maximum string length for the SA command.  It is helpful for
timing to keep the SA command query as short as possible.
```

SB

FUNCTION: Set Bit

DESCRIPTION:
The SB command sets a particular output bit, setting the output to logic 1. The SB and CB (Clear Bit) instructions can be used to control the state of output lines. The SB command can also be used with modbus devices to toggle off-board outputs.
Note 1: When using Modbus devices, the I/O points of the modbus devices are calculated using the following formula:

n = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)
HandleNum is the handle specifier from A,B or C (1 to 3).
Module is the position of the module in the rack from 1 to 16.
BitNum is the I/O point in the module from 1 to 4.

ARGUMENTS: SB n where
n is an integer in the range 0 to 15 decimal.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No
RELATED COMMAND:
"CB" Clear Bit
"OB" Output Bit

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
SB 3          Set output line 3
SB 1          Set output line 1
```

@SIN[n]

FUNCTION: Sine

DESCRIPTION:
Returns the sine of the given angle in degrees

ARGUMENTS: @SIN[n] where
n is a signed number in degrees in the range of -32768 to 32767, with a fractional resolution of 16-bit..

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ASIN[n] Arc sine
@COS[n] Cosine
@ATAN[n] Arc tangent
@ACOS[n] Arc cosine
@TAN[n] Tangent

EXAMPLES:
```
:MG @SIN[0]
0.0000
:MG @SIN[90]
1.0000
:MG @SIN[180]
0.0000
:MG @SIN[270]
-1.0000
:MG @SIN[360]
0.0000
```

SL

FUNCTION: Single Step

DESCRIPTION:
For debugging purposes. Single Step through the program after execution has paused at a breakpoint (BK). Optional argument allows user to specify the number of lines to execute before pausing again. The BK command resumes normal program execution.

ARGUMENTS: SL n where
n is an integer representing the number of lines to execute before pausing again

USAGE:
Default Value 1
In a Program No
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
BK Breakpoint
TR Trace

EXAMPLES:
```
BK 3          Pause at line 3 (the 4th line) in thread 0
BK 5          Continue to line 5
SL          Execute the next line
SL 3          Execute the next 3 lines
BK          Resume normal execution
```

SM

FUNCTION: Subnet Mask

DESCRIPTION:
The SM command assigns a subnet mask to the controller. All packets sent to the controller whose source IP address is not on the subnet will be ignored by the controller. For example, for SM 255, 255, 0, 0 and IA 10, 0, 51, 1, only packets from IP addresses of the form 10.0.xxx.xxx will be accepted.

ARGUMENTS: SM sm0, sm1, sm2, sm3 or SM n where
sm0, sm1, sm2, sm3 are 1 byte numbers (0 to 255) separated by commas and represent the individual fields of the subnet mask.
n is the subnet mask for the controller, which is specified as an integer representing the signed 32 bit number (two's complement).
SM? will return the subnet mask of the controller

USAGE:
Default Value SM 0, 0, 0, 0
In a Program Yes Default Format
Command Line Yes

OPERAND USAGE:
_SM0 contains the IP address representing a 32 bit signed number (Two's complement)

RELATED COMMANDS:
IH Internet Handle
IA IP address

EXAMPLES:
```
SM 255, 255, 255, 255         Ignore all incoming Ethernet packets
SM 0, 0, 0, 0         Process all incoming Ethernet packets
```

@SQR[n]

FUNCTION: Square Root

DESCRIPTION:
Takes the square root of the given number. If the number is negative, the absolute value is taken first.

ARGUMENTS: @SQR[n] where
n is a signed number in the range -2147483648 to 2147483647.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ABS[n] Absolute value

EXAMPLES:
```
:MG @SQR[2]
1.4142
:MG @SQR[-2]
1.4142
```

@TAN[n]

FUNCTION: Tangent

DESCRIPTION:
Returns the tangent of the given angle in degrees

ARGUMENTS: @TAN[n] where
n is a signed number in degrees in the range of -32768 to 32767, with a fractional resolution of 16-bit.

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line Yes

OPERAND USAGE:
RELATED COMMANDS:
@ASIN[n] Arc sine
@COS[n] Cosine
@ATAN[n] Arc tangent
@ACOS[n] Arc cosine
@SIN[n] Sine

EXAMPLES:
```
:MG @TAN[-90]
-2147483647.0000
:MG @TAN[0]
0.0000
:MG @TAN[90]
2147483647.0000
```

TB

FUNCTION: Tell Status Byte

DESCRIPTION:
The TB command returns status information from the RIO as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Executing program
Bit 6	N/A
Bit 5	N/A
Bit 4	N/A
Bit 3	Input interrupt enabled in thread 0
Bit 2	Executing input interrupt routine in thread 0
Bit 1	0 (Reserved)
Bit 0	Echo on

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:
_TB contains the status byte.

RELATED COMMANDS:
EXAMPLES:

```
TB          Tell status information from the RIO board
129          Executing program and echo on  (2^7 + 2^0 = 128 + 1 = 129)
```

TC

FUNCTION: Tell Error Code

DESCRIPTION:
The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the RIO. This command is useful when the RIO board halts execution of a program at a command or when the response to a command is a question mark. Entering the TC command will provide the user with a code as to the reason. After TC has been read, it is set to zero. TC 1 returns the text message as well as the numeric code.

ARGUMENTS: TC n
n=0 returns code only
n=1 returns code and message

CODE	EXPLANATION
1	Unrecognized command
2	Command only valid from program
3	Command not valid in program
4	Operand error
5	Input buffer full
6	Number out of range
9	Variable error
10	Empty program line or undefined label
11	Invalid label or line number
12	Subroutine more than 16 deep
14	EEPROM check sum error
15	EEPROM write error
19	Application strand already executing
25	Variable not given in IN command
50	Not enough fields
51	Question mark not valid
52	Missing " or string too long
53	Error in {}
54	Question mark part of string
55	Missing [or []
56	Array index invalid or out of range
57	Bad function or array
58	Not a valid Command Operand (i.e._GNX)
59	Mismatched parentheses
60	Download error - line too long or too many lines
61	Duplicate or bad label
62	Too many labels
63	IF statement without ENDIF
65	IN command must have a comma
66	Array space full
67	Too many arrays or variables
71	IN only valid in task #0
80	Record mode already running
81	No array or source specified
82	Undefined Array
83	Not a valid number
84	Too many elements
97	Bad Binary Command Format
98	Binary Commands not valid in application program
99	Bad binary command number
120	Bad Ethernet transmit
121	Bad Ethernet packet received
122	Ethernet input buffer overrun
123	TCP lost sync
124	Ethernet handle already in use
125	No ARP response from IP address
126	Closed Ethernet handle
127	Illegal Modbus function code
131	Serial Port Timeout

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_TC contains the value of the error code.

RELATED COMMANDS:
EXAMPLES:

```
:GF32          Bad command
?TC          Tell error code
```

```
1          Unrecognized command
```

#TCPERR

FUNCTION: Ethernet communication error automatic subroutine

DESCRIPTION:
The following error (see TC) occurs when a command such as MG "hello" {EA} is sent to a failed Ethernet connection:
123 TCP lost sync or timeout
This error means that the client on handle A did not respond with a TCP acknowledgement (for example because the Ethernet cable was disconnected). Handle A is closed in this case.
#TCPERR allows the application programmer to run code (for example to reestablish the connection) when error 123 occurs.

ARGUMENTS:
USAGE:
In a Program Yes
Command Line No

OPERAND USAGE:
RELATED COMMANDS:
TC Tell error code
_IA4 Last dropped handle
MG Print message
SA Send ASCII command via Ethernet

EXAMPLES:

```
          #L
            MG {EA} "L"
            WT1000
          JP#L
          #TCPERR
            MG {P1} "TCPERR.  Dropped handle", _IA4
          RE
NOTE: Use RE to end the routine
```

TE

FUNCTION: Tell Error

DESCRIPTION::
This command returns the current error of the analog signal in the control loop. For further details see the Process Control section in the User's Manual.

ARGUMENTS: TE or TEn where
n is A or B
No argument will provide the position error for all channels

USAGE:
Default Value 0
In a Program Yes
Command Line Yes

OPERAND USAGE:
_TEn contains the current position error value for the specified channel.

RELATED COMMANDS:
CL Control Loop update rate
DB Deadband
PS Control Loop Setpoint
AF Analog Feedback Select
AZ Analgo Output Select

EXAMPLES:
```
TE          Return all position errors
:1,-2
Error =_TEA          Sets the variable, Error, with the channel 1 position error
```

TH

FUNCTION: Tell Ethernet Handle

DESCRIPTION:
This command returns a list of data pertaining to the RIO's Ethernet connection. This list begins with the RIO IP address and Ethernet address (physical address), followed by the availability of each of the 3 Ethernet handles.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No
RELATED COMMAND:
"IH"
Internet Handle
"HR" Handle Restore
"WH"
Which Handle

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
:TH
CONTROLLER IP ADDRESS 10,0,51,82 ETHERNET ADDRESS 10-80-3C-10-01-2F  (JH)
IHA TCP PORT 1010 TO IP ADDRESS 10,0,51,87 PORT 1030
IHB TCP PORT 1020 TO IP ADDRESS 10,0,51,87 PORT 1070
IHC AVAILABLE
```

TI

FUNCTION: Tell Inputs

DESCRIPTION:
This command returns the state of inputs in banks of 8. Response is a decimal number which when converted to binary represents the status of the digital inputs for the specified bank. TI0 specifies inputs 0-7 (bank 0) and TI1 specifies inputs 8-15 (bank 1) - the response will be a number from 0 to 255.

ARGUMENTS: TI n where
n ranges from 0 thru 1.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_TIn contains the status of the input module for bank n.
Note: The operand can be masked to return only specified bit information - see section on Bitwise operations.

RELATED COMMANDS:
EXAMPLES:

```
TI1          Tell input state on bank 1
8          Bit 3 on slot 1 is high, others low
TI0
0          All inputs on bank 0 low
Input =_TI1          Sets the variable, Input, with the TI1 value
:Input=?
8.0000
```

TIME*

FUNCTION: Time Operand (Keyword)

DESCRIPTION:
*The TIME operand contains the value of the intenal free running, real time clock. The operand TIME will increase by 1 count every millisecond.
The clock is reset to 0 with a standard reset or a master reset.
The keyword, TIME, does not require an underscore (_) as with the other operands.

ARGUMENTS:
USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
MG TIME          Display the value of the internal clock
T1=TIME          Sets the variable, T1, with the TIME value
```

TR

FUNCTION: Trace

DESCRIPTION:
The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

ARGUMENTS: TR n where
n = 0 or 1
0 disables function
1 enables function

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

TZ

FUNCTION: Tell I/O Configuration

DESCRIPTION:
This command returns a list of information pertaining to the status of the RIO's I/O. On each line, the information starts with the I/O block, the corresponding I/O number range, the configuration (as inputs or outputs), and the values (255 for all 8 output bits high etc.)

ARGUMENTS: TZ

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No
RELATED COMMAND:
"TH" Tell Ethernet Handle
"TQ" Tell Thread Execution

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
:TZ
Block 0 (7-0) Inputs - value 255 (1111_1111)
Block 1 (15-8) Inputs - value 255 (1111_1111)
Block 0 (7-0) Outputs - value 0 (0000_0000)
Block 1 (15-8) Outputs - value 0 (0000_0000)
Analog Inputs(7-0)  0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
Analog Outputs(7-0) 0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
```

UL

FUNCTION: Upload

DESCRIPTION:
The UL command transfers data from the RIO to a host computer. Programs are sent without line numbers. In the Galil software, the UL command is not necessary because the UL command is handled by the graphical interface (Upload Program). In a terminal utility such as HyperTerminal or Telnet, the UL command will bring the uploaded program to screen. From there, the user can copy it and save it to a file. In Hyper Terminal, the UL command will be followed by a Z or a \ as an end of Text marker.

ARGUMENTS: None

USAGE:
In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
When used as an operand, _UL gives the number of available variables. The total number of variables is 126.
RELATED COMMAND:
"DL" Download

RELATED COMMANDS:
EXAMPLES:

```
UL;          Begin upload
#A;'          Line 0
SB1;'           This is an Example Line 1
CB1;'           Program         Line 2
EN;'          Line 3
{cntrl}Z          Terminator
```

VF

FUNCTION: Variable Format

DESCRIPTION:
The VF command formats the number of digits to be displayed when interrogating the RIO board. If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n where
m and n are unsigned numbers in the range 0<m<n<4.
m represents the number of digits before the decimal point. A negative m specifies hexadecimal format. When in hexadecimal, the string will be preceded by a $ and Hex numbers are displayed as 2's complement with the first bit used to signify the sign.
n represents the number of digits after the decimal point.
m = ? Returns the value of the format for variables and arrays.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_VF contains the value of the format for variables and arrays.

RELATED COMMANDS:
EXAMPLES:

```
VF 5.3          Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0          Sets 8 digits of integers and no fractions
VF -4.0          Specify hexadecimal format with 4 bytes to the left of the
decimal
```

WH

FUNCTION: Which Handle

DESCRIPTION:
The WH command is used to identify the handle in which the command is executed on. The command returns IHA, IHB, or IHC to indicate on which handle the command was executed on. The command returns RS232 if using serial communication.

ARGUMENTS: None

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes
RELATED COMMAND:
"TH" Tell Handle
"IH" Internet Handle

OPERAND USAGE:
_WH contains the numeric representation of the handle in which a command is executed. Handles A - C are indicated by the value 0 - 2, while a -1 indicates the serial port.

RELATED COMMANDS:
EXAMPLES:

```
:WH          Request handle identification
IHB          Command executed in handle B
:WH          Request handle identification
RS232          Command executed in RS232 port
```

WT

FUNCTION: Wait

DESCRIPTION:
The WT command is a trippoint used to time events. After this command is executed, the RIO board will wait for the number of milliseconds specified before executing the next command. The units of the Wait command are in milliseconds.

ARGUMENTS: WT n where
n is an integer in the range 0 to 2 Billion

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

OPERAND USAGE:
RELATED COMMANDS:
EXAMPLES:

```
Assume that 10 seconds after an input goes high, send message to user.
#A                  Program A
AI3                  After input 3 goes high
WT 10000          Wait 10 seconds
MG "RELAY ON"          Print message
EN                  End Program
```

XQ

FUNCTION: Execute Program

DESCRIPTION:
The XQ command begins execution of a program residing in the program memory of the RIO board. Execution will start at the label or line number specified. Up to 4 programs may be executed simultaneously to perform multitasking.

ARGUMENTS: XQ #A,n XQm,n where
A is a program name of up to seven characters
m is a line number
n is the thread number (0 thru 3) for multitasking
NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

OPERAND USAGE:
_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

RELATED COMMANDS:
"HX"
Halt execution

EXAMPLES:
```
XQ #Apple,0         Start execution at label Apple, thread zero
XQ #data,1         Start execution at label data, thread one
XQ 0         Start execution at line 0
```

ZC

FUNCTION: User Variable, ZC

DESCRIPTION:
ZC sets the first user variable. This variable provides a method for specific board information to be passed using the data record.

ARGUMENTS: ZCn where
n can be a number, RIO board operand, variable, mathematical function, or string; The range for numeric values is from -2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.
NOTE: If n is a '?,' the decimal value of ZC will be returned.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_ZC contains the decimal value of the user variable.

RELATED COMMANDS:
"ZD" Set second user variable

EXAMPLES:
```
ZC 2343          Sets the first user variable to a number (2343)
```

ZD

FUNCTION: User Variable, ZD

DESCRIPTION:
ZD sets the second user variable. This variable provides a method for specific board information to be passed using the data record.

ARGUMENTS: ZDn where
n can be a number, RIO board operand, variable, mathematical function, or string; The range for numeric values is from -2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.
NOTE: If n is a '?,' the decimal value of ZD will be returned.

USAGE:
In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_ZD contains the decimal value of the user variable.

RELATED COMMANDS:
"ZC" Set first user variable

EXAMPLES:
```
ZD "INPT"          Sets the second user variable to the string "INPT"
```

ZS

FUNCTION: Zero Subroutine Stack

DESCRIPTION:
The ZS command is only valid in an application program and is used to avoid returning from an input interrupt. ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the 'jump to subroutine, JS,' into a jump, JP. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use the II command again.

ARGUMENTS: ZS n where
0 returns stack to original condition
1 eliminates one return on stack

USAGE:
In a Program Yes
Command Line No
Can be Interrogated Yes
Used as an Operand Yes

OPERAND USAGE:
_ZSn contains the stack level for the specified thread where n = 0 thru 3. The response, an integer between zero and sixteen, indicates zero for beginning condition and sixteen for the deepest value.

RELATED COMMANDS:
EXAMPLES:

```
II0,1,7                  Input Interrupt on 7
#A;JP #A;EN          Main program
#ININT0                  Input Interrupt
MG "INTERRUPT"          Print message
S=_ZS                    Interrogate stack before ZS
S=                   Print stack
ZS                   Zero stack
S=_ZS                    Interrogate stack after ZS
S=                   Print stack
EN
```